ROBUST TEXT PROCESSING IN AUTOMATED INFORMATION RETRIEVAL

Tomek Strzalkowski
Courant Institute of Mathematical Sciences
New York University
715 Broadway, rm. 704
New York, NY 10003
tomek@cs.nyu.edu

ABSTRACT

This paper outlines a prototype text retrieval system which uses relatively advanced natural language processing techniques in order to enhance the effectiveness of statistical document retrieval. The backbone of our system is a traditional retrieval engine which builds inverted index files from pre-processed documents, and then searches and ranks the documents in response to user queries. Natural language processing is used to (1) preprocess the documents in order to extract contents-carrying terms, (2) discover inter-term dependencies and build a conceptual hierarchy specific to the database domain, and (3) process user's natural language requests into effective search queries. The basic assumption of this design is that term-based representation of contents is in principle sufficient to build an effective if not optimal search query out of any user's request. This has been confirmed by an experiment that compared effectiveness of expert-user prepared queries with those derived automatically from an initial narrative information request. In this paper we show that large-scale natural language processing (hundreds of millions of words and more) is not only required for a better retrieval, but it is also doable, given appropriate resources. We report on selected preliminary results of experiments with 500 MByte database of Wall Street Journal articles, as well as some earlier results with a smaller document collection.

INTRODUCTION

A typical information retrieval (IR) task is to select documents from a database in response to a user's query, and rank these documents according to relevance. This has been usually accomplished using statistical methods (often coupled with manual encoding) that (a) select terms (words, phrases, and other units) from documents that are deemed to best represent their contents, and (b) create an inverted index file (or files) that provide and easy access to documents containing these terms. An important issue here is that of finding an appropriate combination of term weights which would reflect each term's relative contribution to the information contents of the document. Among many possible weighting schemes the *inverted document frequency* (idf) has come to be recognized as universally applicable across variety of different text collections.

Once the index is created, the search process will attempt to match a preprocessed user query (or queries) against representations of documents in each case determining a degree of relevance between the two which depends upon the number and types of matching terms. Although many sophisticated search and matching methods are available, the crucial problem remains to be that of an adequate representation of contents for both the documents and the queries.

The simplest word-based representations of contents are usually inadequate since single words are rarely specific enough for accurate discrimination, and their grouping is often accidental. A better method is to identify groups of words that create meaningful *phrases*, especially if these phrases denote important concepts in database domain. For example, *joint venture* is an important term in Wall Street Journal (WSJ henceforth) database, while neither *joint* nor *venture* are important by themselves. In the retrieval experiments with the WSJ database, we noticed that both *joint* and *venture* were dropped from the list of terms by the system because their idf weights were too low. In large databases, such as TIPSTER/TREC, the use of phrasal terms is not just desirable, it becomes necessary.

The question thus becomes, how to identify the correct phrases in the text? Both statistical and syntactic methods were used before with only limited success. Statistical methods based on word co-occurrences and mutual information are prone to high error rates, turning out many unwanted associations. Syntactic methods suffered from low quality of generated parse structures that could be attributed to limited coverage grammars and the lack of adequate lexicons. In fact, the difficulties encountered in applying computational linguistics technologies to text processing have contributed to a wide-spread belief that

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1993		**00-00-1993 to 00-00-1993**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Robust Text Processing in Automated Information Retrieval	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Computer Science ,New York University,715 Broadway,New York,NY,10003**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**11**	

automated natural language processing may not be suitable in IR. These difficulties included inefficiency, lack of robustness, and prohibitive cost of manual effort required to build lexicons and knowledge bases for each new text domain. On the other hand, while numerous experiments did not establish the usefulness of linguistic methods in IR, they cannot be considered conclusive because of their limited scale. [1]

The rapid progress in Computational Linguistics over the last few years has changed this equation in various ways. First of all, large-scale resources became available: on-line lexicons, including Oxford Advanced Learner's Dictionary (OALD), Longman Dictionary of Contemporary English (LDOCE), Webster's Dictionary, Oxford English Dictionary, Collins Dictionary, and others, as well as large text corpora, many of which can now be obtained for research purposes. Robust text-oriented software tools have been built, including part of speech taggers (stochastic and otherwise), and fast parsers capable of processing text at speeds of 4200 words per minute or more (e.g., TTP parser developed by the author). While many of the fast parsers are not very accurate (they are usually partial analyzers by design),[2] some, like TTP, perform in fact no worse than standard full-analysis parsers which are many times slower and far less robust. [3]

An accurate syntactic analysis is an essential prerequisite for term selection, but it is by no means sufficient. Syntactic parsing of the database contents is usually attempted in order to extract linguistically motivated phrases, which presumably are better indicators of contents than "statistical phrases" where words are grouped solely on the basis of physical proximity (e.g., "college junior" is not the same as "junior college"). However, creation of such compound terms makes term matching process more complex since in addition to the usual problems of synonymy and subsumption, one must deal with their structure (e.g., "college junior" is the same as "junior in college"). In order to deal with structure, parser's output needs to be "normalized" or "regularized" so that complex terms with the same or closely related meanings would indeed receive matching representations. This goal has been achieved to a certain extent in the present work. As it will be discussed in more detail below, indexing terms were selected from among head-modifier pairs extracted from predicate-argument representations of sentences.

The next important task is to achieve normalization across diferent terms with close or related meaning. This can be accomplished by discovering various semantic relationships among words and phrases, such as synonymy and subsumption. For example, the term *natural language* can be considered, in certain domains at least, to subsume any term denoting a specific human language, such as *English*. Therefore, a query containing the former may be expected to retrieve documents containing the latter. The system presented here computes term associations from text on word and fixed phrase level and then uses these associations in query expansion. A fairly primitive filter is employed to separate synonymy and subsumption relationships from others including antonymy and complementation, some of which are strongly domain-dependent. This process has led to an increased retrieval precision in experiments with smaller and more cohesive collections (CACM-3204).

In the following sections we present an overview of our system, with the emphasis on its text-processing components. We would like to point out here that the system is completely automated, i.e., all the processing steps, those performed by the statistical core, and these performed by the natural language processing components, are done automatically, and no human intervention or manual encoding is required.

OVERALL DESIGN

Our information retrieval system consists of a traditional statistical backbone (NIST's PRISE system; Harman and Candela, 1989) augmented with various natural language processing components that assist the system in database processing (stemming, indexing, word and phrase clustering, selectional restrictions), and translate a user's information request into an effective query. This design is a careful compromise between purely statistical non-linguistic approaches and those requiring rather accomplished (and expensive) semantic analysis of data, often referred to as 'conceptual retrieval'.

In our system the database text is first processed with a fast syntactic parser. Subsequently certain types of phrases are extracted from the parse

[1] Standard IR benchmark collections are statistically too small and the experiments can easily produce counterintuitive results. For example, Cranfield collection is only approx. 180,000 English words, while CACM-3204 collection is approx. 200,000 words.

[2] Partial parsing is usually fast enough, but it also generates noisy data: as many as 50% of all generated phrases could be incorrect (Lewis and Croft, 1990).

[3] TTP has been shown to produce parse structures which are no worse in recall, precision and crossing rate than those generated by full-scale linguistic parsers when compared to hand-coded Treebank parse trees.

trees and used as compound indexing terms in addition to single-word terms. The extracted phrases are statistically analyzed as syntactic contexts in order to discover a variety of similarity links between smaller subphrases and words occurring in them. A further filtering process maps these similarity links onto semantic relations (generalization, specialization, synonymy, etc.) after which they are used to transform user's request into a search query.

The user's natural language request is also parsed, and all indexing terms occurring in them are identified. Certain highly ambiguous, usually single-word terms may be dropped, provided that they also occur as elements in some compound terms. At the same time, other terms may be added, namely those which are linked to some query term through admissible similarity relations. For example, "unlawful activity" is added to a query containing the compound term "illegal activity" via a synonymy link between "illegal" and "unlawful". After the final query is constructed, the database search follows, and a ranked list of documents is returned.

The purpose of this elaborate linguistic processing is to create a better representation of documents and to generate best possible queries out of user's initial requests. Despite limitations of term-and-weight type representation (or boolean versions thereof), very good queries can be produced by human experts. In order to imitate an expert, the system must be able to learn about its database, in particular about various correlations among index terms.

FAST PARSING WITH TTP PARSER

TTP (Tagged Text Parser) is based on the Linguistic String Grammar developed by Sager (1981). The parser currently encompasses some 400 grammar productions, but it is by no means complete. The parser's output is a regularized parse tree representation of each sentence, that is, a representation that reflects the sentence's logical predicate-argument structure. For example, logical subject and logical object are identified in both passive and active sentences, and noun phrases are organized around their head elements. The significance of this representation will be discussed below. The parser is equipped with a powerful skip-and-fit recovery mechanism that allows it to operate effectively in the face of ill-formed input or under a severe time pressure. In the runs with approximately 83 million words of TREC's Wall Street Journal texts,[4] the parser's speed averaged between 0.3 and 0.5 seconds per sentence, or up to 4200 words per minute, on a Sun's SparcStation-2.

TTP is a full grammar parser, and initially, it attempts to generate a complete analysis for each sentence. However, unlike an ordinary parser, it has a built-in timer which regulates the amount of time allowed for parsing any one sentence. If a parse is not returned before the allotted time elapses, the parser enters the skip-and-fit mode in which it will try to "fit" the parse. While in the skip-and-fit mode, the parser will attempt to forcibly reduce incomplete constituents, possibly skipping portions of input in order to restart processing at a next unattempted constituent. In other words, the parser will favor reduction to backtracking while in the skip-and-fit mode. The result of this strategy is an approximate parse, partially fitted using top-down predictions. The fragments skipped in the first pass are not thrown out, instead they are analyzed by a simple phrasal parser that looks for noun phrases and relative clauses and then attaches the recovered material to the main parse structure. As an illustration, consider the following sentence taken from the CACM-3204 corpus:

> The method is illustrated by the automatic construction of both recursive and iterative programs operating on natural numbers, lists, and trees, in order to construct a program satisfying certain specifications *a theorem induced by those specifications is proved*, and the desired program is extracted from the proof.

The italicized fragment is likely to cause additional complications in parsing this lengthy string, and the parser may be better off ignoring this fragment altogether. To do so successfully, the parser must close the currently open constituent (i.e., reduce *a program satisfying certain specifications* to *NP*), and possibly a few of its parent constituents, removing corresponding productions from further consideration, until an appropriate production is reactivated. In this case, TTP may force the following reductions: $SI \rightarrow$ *to V NP*; $SA \rightarrow SI$; $S \rightarrow$ *NP V NP SA*, until the production $S \rightarrow S$ *and* S is reached. Next, the parser skips input to find *and*, and resumes normal processing.

As may be expected, the skip-and-fit strategy will only be effective if the input skipping can be performed with a degree of determinism. This means that most of the lexical level ambiguity must be removed from the input text, prior to parsing. We achieve this using a stochastic parts of speech tagger to preprocess the text. Full details of the parser can be found in (Strzalkowski, 1992).

[4] Approximately 0.5 GBytes of text, over 4 million sentences.

PART OF SPEECH TAGGER

One way of dealing with lexical ambiguity is to use a tagger to preprocess the input marking each word with a tag that indicates its syntactic categorization: a part of speech with selected morphological features such as number, tense, mode, case and degree. The following are tagged sentences from the CACM-3204 collection: [5]

> The/*dt* paper/*nn* presents/*vbz* a/*dt* proposal/*nn* for/*in* structured/*vbn* representation/*nn* of/*in* multiprogramming/*vbg* in/*in* a/*dt* high/*jj* level/*nn* language/*nn* ./*per*

> The/*dt* notation/*nn* used/*vbn* explicitly/*rb* associates/*vbz* a/*dt* data/*nns* structure/*nn* shared/*vbn* by/*in* concurrent/*jj* processes/*nns* with/*in* operations/*nns* defined/*vbn* on/*in* it/*pp* ./*per*

The tags are understood as follows: dt - determiner, nn - singular noun, nns - plural noun, in - preposition, jj - adjective, vbz - verb in present tense third person singular, to - particle "to", vbg - present participle, vbn - past participle, vbd - past tense verb, vb - infinitive verb, cc - coordinate conjunction.

Tagging of the input text substantially reduces the search space of a top-down parser since it resolves most of the lexical level ambiguities. In the examples above, tagging of *presents* as "vbz" in the first sentence cuts off a potentially long and costly "garden path" with *presents* as a plural noun followed by a headless relative clause starting with *(that) a proposal* .... In the second sentence, tagging resolves ambiguity of *used* (vbn vs. vbd), and *associates* (vbz vs. nns). Perhaps more importantly, elimination of word-level lexical ambiguity allows the parser to make projection about the input which is yet to be parsed, using a simple lookahead; in particular, phrase boundaries can be determined with a degree of confidence (Church, 1988). This latter property is critical for implementing skip-and-fit recovery technique outlined in the previous section.

Tagging of input also helps to reduce the number of parse structures that can be assigned to a sentence, decreases the demand for consulting of the dictionary, and simplifies dealing with unknown words. Since every item in the sentence is assigned a tag, so are the words for which we have no entry in the lexicon. Many of these words will be tagged as "np" (proper noun), however, the surrounding tags may force other selections. In the following example, *chinese*, which does not appear in the dictionary, is tagged as "jj":[6]

> this/*dt* paper/*nn* dates/*vbz* back/*rb* the/*dt* genesis/*nn* of/*in* binary/*jj* conception/*nn* circa/*in* 5000/*cd* years/*nns* ago/*rb* ,/*com* as/*rb* derived/*vbn* by/*in* the/*dt* chinese/*jj* ancients/*nns* ./*per*

WORD SUFFIX TRIMMER

Word stemming has been an effective way of improving document recall since it reduces words to their common morphological root, thus allowing more successful matches. On the other hand, stemming tends to decrease retrieval precision, if care is not taken to prevent situations where otherwise unrelated words are reduced to the same stem. In our system we replaced a traditional morphological stemmer with a conservative dictionary-assisted suffix trimmer. [7] The suffix trimmer performs essentially two tasks: (1) it reduces inflected word forms to their root forms as specified in the dictionary, and (2) it converts nominalized verb forms (e.g., "implementation", "storage") to the root forms of corresponding verbs (i.e., "implement", "store"). This is accomplished by removing a standard suffix, e.g., "stor+age", replacing it with a standard root ending ("+e"), and checking the newly created word against the dictionary, i.e., we check whether the new root ("store") is indeed a legal word, and whether the original root ("storage") is defined using the new root ("store") or one of its standard inflectional forms (e.g., "storing"). For example, the following definitions are excerpted from the *Oxford Advanced Learner's Dictionary* (OALD):

> **storage** *n* [U] (space used for, money paid for) the storing of goods ...
> **diversion** *n* [U] diverting ...
> **procession** *n* [C] number of persons, vehicles, etc moving forward and following each other in an orderly way.

Therefore, we can reduce "diversion" to "divert" by removing the suffix "+sion" and adding root form suffix "+t". On the other hand, "process+ion" is not reduced to "process".

Earlier experiments with CACM-3204 collection showed an improvement in retrieval precision by 6% to 8% over the base system equipped with a standard morphological stemmer (the SMART stemmer).

---

[5] Tagged using the 35-tag Penn Treebank Tagset created at the University of Pennsylvania.

[6] We use the machine readable version of the Oxford Advanced Learner's Dictionary (OALD).

[7] Dealing with prefixes is a more complicated matter, since they may have quite strong effect upon the meaning of the resulting term, e.g., *un-* usually introduces explicit negation.

HEAD-MODIFIER STRUCTURES

Syntactic phrases extracted from TTP parse trees are head-modifier pairs. The head in such a pair is a central element of a phrase (main verb, main noun, etc.), while the modifier is one of the adjunct arguments of the head. In the TREC experiments reported here we extracted head-modifier word and fixed-phrase pairs only. While TREC WSJ database is large enough to warrant generation of larger compounds, we were in no position to verify their effectiveness in indexing. This was largely because of the tight schedule, but also because of rapidly escalating complexity of the indexing process: even with 2-word phrases, compound terms accounted for nearly 96% of all index entries, in other words, including 2-word phrases has increased the index size 25 times!

Let us consider a specific example from WSJ database:

> The former Soviet president has been a local hero ever since a Russian tank invaded Wisconsin.

The tagged sentence is given below, followed by the regularized parse structure generated by TTP, given in Figure 1.

> The/*dt* former/*jj* Soviet/*jj* president/*nn* has/*vbz* been/*vbn* a/*dt* local/*jj* hero/*nn* ever/*rb* since/*in* a/*dt* Russian/*jj* tank/*nn* invaded/*vbd* Wisconsin/*np* ./*per*

It should be noted that the parser's output is a predicate-argument structure centered around main elements of various phrases. In Figure 1, BE is the main predicate (modified by HAVE) with 2 arguments (*subject, object*) and 2 adjuncts (*adv, sub_ord*). INVADE is the predicate in the subordinate clause with 2 arguments (*subject, object*). The subject of BE is a noun phrase with PRESIDENT as the head element, two modifiers (FORMER, SOVIET) and a determiner (THE). From this structure, we extract head-modifier pairs that become candidates for compound terms. The following types of pairs are considered: (1) a head noun and its left adjective or noun adjunct, (2) a head noun and the head of its right adjunct, (3) the main verb of a clause and the head of its object phrase, and (4) the head of the subject phrase and the main verb. These types of pairs account for most of the syntactic variants for relating two words (or simple phrases) into pairs carrying compatible semantic content. For example, the pair *retrieve+information* will be extracted from any of the following fragments: *information retrieval system; retrieval of information from databases*; and *information that can be retrieved by a user-controlled interactive search process*. In the example at hand, the following head-modifier pairs are extracted (pairs containing low-contents elements,

```
[assert
 [[perf [HAVE]]
  [[verb [BE]]
   [subject
    [np
     [n PRESIDENT]
     [t_pos THE]
     [adj [FORMER]]
     [adj [SOVIET]]]]
   [object
    [np
     [n HERO]
     [t_pos A]
     [adj [LOCAL]]]]
   [adv EVER]
   [sub_ord
    [SINCE
     [[verb [INVADE]]
      [subject
       [np
        [n TANK]
        [t_pos A]
        [adj [RUSSIAN]]]]
      [object
       [np
        [name [WISCONSIN]]]]]]]]]]
```

Figure 1. Predicate-argument parse structure.

such as BE and FORMER, or names, such as WISCONSIN, will be later discarded):

> [PRESIDENT,BE]
> [PRESIDENT,FORMER]
> [PRESIDENT,SOVIET]
> [BE,HERO]
> [HERO,LOCAL]
> [TANK,INVADE]
> [TANK,RUSSIAN]
> [INVADE,WISCONSIN]

We may note that the three-word phrase *former Soviet president* has been broken into two pairs *former president* and *Soviet president*, both of which denote things that are potentially quite different from what the original phrase refers to, and this fact may have potentially negative effect on retrieval precision. This is one place where a longer phrase appears more appropriate. The representation of this sentence may therefore contain the following terms:

> PRESIDENT, SOVIET, PRESIDENT+SOVIET, PRESIDENT+FORMER, HERO, HERO+LOCAL, INVADE, TANK, TANK+INVADE, TANK+RUSSIAN, RUSSIAN, INVADE+WISCONSIN, WISCONSIN.

The particular way of interpreting syntactic contexts was dictated, to some degree at least, by statistical considerations. Our original experiments

were performed on a relatively small collection (CACM-3204), and therefore we combined pairs obtained from different syntactic relations (e.g., verb-object, subject-verb, noun-adjunct, etc.) in order to increase frequencies of some associations. This became largely unnecessary in a large collection such as TIPSTER, but we had no means to test alternative options, and thus decided to stay with the original. It should not be difficult to see that this was a compromise solution, since many important distinctions were potentially lost, and strong associations could be produced where there weren't any. A way to improve things is to consider different syntactic relations independently, perhaps as independent sources of evidence that could lend support (or not) to certain term similarity predictions. We have already started testing this option.

One difficulty in obtaining head-modifier pairs of highest accuracy is the notorious ambiguity of nominal compounds. For example, the phrase *natural language processing* should generate *language+natural* and *processing+language*, while *dynamic information processing* is expected to yield *processing+dynamic* and *processing+information*. A still another case is *executive vice president* where the association *president+executive* may be stretching things a bit too far. Since our parser has no knowledge about the text domain, and uses no semantic preferences, it does not attempt to guess any internal associations within such phrases. Instead, this task is passed to the pair extractor module which processes ambiguous parse structures in two phases. In phase one, all and only unambiguous head-modifier pairs are extracted, and the frequencies of their occurrences are recorded. In phase two, frequency information about pairs generated in the first pass is used to form associations from ambiguous structures. For example, if *language+natural* has occurred unambiguously a number times in contexts such as *parser for natural language*, while *processing+natural* has occurred significantly fewer times or perhaps none at all, then we will prefer the former association as valid.

TERM CORRELATIONS FROM TEXT

Head-modifier pairs form compound terms used in database indexing. They also serve as occurrence contexts for smaller terms, including single-word terms. If two terms tend to be modified with a number of common modifiers and otherwise appear in few distinct contexts, we assign them a similarity coefficient, a real number between 0 and 1. The similarity is determined by comparing distribution characteristics for both terms within the corpus: how much information contents do they carry, do their information contribution over contexts vary greatly, are the common contexts in which these terms occur specific enough? In general we will credit high-contents terms appearing in identical contexts, especially if these contexts are not too commonplace.[8] The relative similarity between two words x_1 and x_2 can be obtained using the following formula (α is a large constant): [9]

$$SIM(x_1,x_2) = log\,(\alpha \sum_y sim_y(x_1,x_2))$$

where

$$sim_y(x_1,x_2) = MIN\,(IC\,(x_1,[x_1,y]),IC\,(x_2,[x_2,y])) * MIN\,(IC\,(y,[x_1,y]),IC\,(y,[x_2,y]))$$

and IC is the Information Contribution measure indicating the strength of word pairings, and defined as

$$IC\,(x,[x,y]) = \frac{f_{x,y}}{n_x+d_x-1}$$

where $f_{x,y}$ is the absolute frequency of pair [x,y] in the corpus, n_x is the frequency of term x at the head position, and d_x is a dispersion parameter understood as the number of distinct syntactic contexts in which term x is found. The similarity function is further normalized with respect to $SIM\,(x_1,x_1)$. Example similarities are listed in Table 1.

We also considered a term clustering option which, unlike the similarity formula above, produces clusters of related words and phrases, but will not generate uniform term similarity ranking across clusters. We used a variant of weighted Tanimoto's measure described in (Grefenstette, 1992):

$$SIM\,(x_1,x_2) = \frac{\sum_{att} MIN\,(W\,([x,att]),W\,([y,att])}{\sum_{att} MAX\,(W\,([x,att]),W\,([y,att])}$$

with

$$W\,([x,y]) = GW\,(x)*log\,(f_{x,y})$$

$$GW\,(x) = 1 - \sum_y \left[\frac{\frac{f_{x,y}}{n_y} * \log\left[\frac{f_{x,y}}{n_y}\right]}{log\,(N)} \right]$$

[8] It would not be appropriate to predict similarity between *language* and *logarithm* on the basis of their co-occurrence with *natural*.

[9] This was inspired by a formula used by Hindle (1990).

Sample clusters obtained from approx. 100 MByte (17 million words) sample of WSJ are given in Table 2.

In order to generate better similarities and clusters, we require that words x_1 and x_2 appear in at least M *distinct* common contexts, where a common context is a couple of pairs $[x_1,y]$ and $[x_2,y]$, or $[y,x_1]$ and $[y,x_2]$ such that they each occurred at least twice. Thus, *banana* and *Baltic* will not be considered for similarity relation on the basis of their occurrences in the common context of *republic*, no matter how frequent, unless there is another such common context comparably frequent (there wasn't any in TREC WSJ database). For smaller or narrow domain databases M=2 is usually sufficient. For large databases covering rather diverse subject matter, like TIPSTER or even WSJ, we used $M \geq 3$.[10]

It may be worth pointing out that the similarities are calculated using term co-occurrences in syntactic rather than in document-size contexts, the latter being the usual practice in non-linguistic clustering (e.g., Sparck Jones and Barber, 1971; Crouch, 1988; Lewis and Croft, 1990). Although the two methods of term clustering may be considered mutually complementary in certain situations, we believe that more and stronger associations can be obtained through syntactic-context clustering, given sufficient amount of data and a reasonably accurate syntactic parser.[11]

QUERY EXPANSION

Similarity relations are used to expand user queries with new terms, in an attempt to make the final search query more comprehensive (adding synonyms) and/or more pointed (adding specializations).[12] It follows that not all similarity relations will be equally useful in query expansion, for instance, complementary and antonymous relations like the one between *Australian* and *Canadian*, or *accept* and *reject* may actually harm system's performance, since we may end up retrieving many irrelevant documents. Similarly, the effectiveness of a query containing *vitamin* is likely to diminish if we add a similar but far more general term such as *acid*. On the other hand, database search is likely to miss relevant documents if we overlook the fact that *fortran* is a *programming language*, or that *infant* is a *baby* and *baby* is a *child*. We noted that an average set of similarities generated from a text corpus contains about as many "good" relations (synonymy, specialization) as "bad" relations (antonymy, complementation, generalization), as seen from the query expansion viewpoint. Therefore any attempt to separate these two classes and to increase the proportion of "good" relations should result in improved retrieval. This has indeed been confirmed in our experiments where a relatively crude filter has visibly increased retrieval precision.

In order to create an appropriate filter, we devised a global term specificity measure (GTS) which is calculated for each term across all contexts in which it occurs. The general philosophy here is that a more specific word/phrase would have a more limited use, i.e., a more specific term would appear in fewer *distinct* contexts. In this respect, GTS is similar to the standard *inverted document frequency (idf)* measure except that term frequency is measured over syntactic units rather than document size units.[13] Terms with higher GTS values are generally considered more specific, but the specificity comparison is only meaningful for terms which are already known to be similar. The new function is calculated according to the following formula:

$$GTS(w) = \begin{cases} IC_L(w) * IC_R(w) & \text{if } \textit{both exist} \\ IC_R(w) & \text{if } \textit{only } IC_R(w) \textit{ exists} \\ 0 & \textit{otherwise} \end{cases}$$

where (with n_w, $d_w > 0$):

$$IC_L(w) = IC([w,\_]) = \frac{n_w}{d_w(n_w+d_w-1)}$$

$$IC_R(w) = IC([\_,w]) = \frac{n_w}{d_w(n_w+d_w-1)}$$

For any two terms w_1 and w_2, and a constant $\delta > 1$, if $GTS(w_2) \geq \delta * GTS(w_1)$ then w_2 is considered more specific than w_1. In addition, if

[10] For example *banana* and *Dominican* were found to have two common contexts: *republic* and *plant*, although this second occurred in apparently different senses in *Dominican plant* and *banana plant*.

[11] Non-syntactic contexts cross sentence boundaries with no fuss, which is helpful with short, succinct documents (such as CACM abstracts), but less so with longer texts; see also (Grishman et al., 1986).

[12] Query expansion (in the sense considered here, though not quite in the same way) has been used in information retrieval research before (e.g., Sparck Jones and Tait, 1984; Harman, 1988), usually with mixed results. An alternative is to use term clusters to create new terms, "metaterms", and use them to index the database instead (e.g., Crouch, 1988; Lewis and Croft, 1990). We found that the query expansion approach gives the system more flexibility, for instance, by making room for hypertext-style topic exploration via user feedback.

[13] We believe that measuring term specificity over document-size contexts (e.g., Sparck Jones, 1972) may not be appropriate in this case. In particular, syntax-based contexts allow for processing texts without any internal document structure.

$SIM_{norm}(w_1, w_2) = \sigma > \theta,$

where θ is an empirically established threshold, then w_2 can be added to the query containing term w_1 with weight σ.[14] For example, the following were obtained from TREC WSJ training database:

GTS (*child*)	= 0.000001
GTS (*baby*)	= 0.000013
GTS (*infant*)	= 0.000055

with

SIM (*child*,*infant*)	= 0.131381
SIM (*baby*,*child*)	= 0.183064
SIM (*baby*,*infant*)	= 0.323121

Therefore both *baby* and *infant* can be used to specialize *child*. With this filter, the relationship between *baby* and *infant* had to be discarded, as we are unable to tell synonymous or near synonymous relationships from those which are primarily complementary, e.g., *man* and *woman*.

SUMMARY OF RESULTS

We have processed the total of 500 MBytes of articles from Wall Street Journal section of TREC database. Retrieval experiments involved 50 user information requests (topics) (TREC topics 51-100) consisting of several fields that included both text and user supplied keywords. A typical topic is shown below:

<top>
<head> Tipster Topic Description
<num> Number: 059
<dom> Domain: Environment
<title> Topic: Weather Related Fatalities

<desc> Description:
Document will report a type of weather event which has directly caused at least one fatality in some location.

<narr> Narrative:
A relevant document will include the number of people killed and injured by the weather event, as well as reporting the type of weather event and the location of the event.

<con> Concept(s):
1. lightning, avalanche, tornado, typhoon, hurricane, heat, heat wave, flood, snow, rain, downpour, blizzard, storm, freezing temperatures
2. dead, killed, fatal, death, fatality, victim
3. NOT man-made disasters, NOT war-induced famine
4. NOT earthquakes, NOT volcanic eruptions
</top>

Note that this topic actually consists of two different statements of the same query: the natural language specification consisting of <desc> and <narr> fields, and an expert-selected list of key terms which are often far more informative than the narrative part. Results obtained for queries using text fields only and those involving both text and keyword fields are reported separately. Further experiments have suggested that natural language processing impact is significant but may be severely limited by the expressiveness of the term-based representation. Since the <con> field is considered the expert-user's rendering of the 'optimal' search query, our system is able to discover much of it from a less complete specification in the text section of the request via query expansion. In fact, we noted that the recall/precision gap between automatically generated queries and those supplied by the user was largely closed when NLP was used. Moreover, even with the keyword field included in the query along with other fields, NLP's impact on the system's performance is still noticeable.

Other results on the impact of different fields in TREC topics on the final recall/precision results were reported by Broglio and Croft (1993) at the ARPA HLT workshop, although text-only runs were not included. One of the most striking observations they have made is that the narrative field is entirely disposable, and moreover that its inclusion in the query actually hurts the system's performance. It has to be pointed out, however, that they do little language processing.[15]

Summary statistics for these runs are shown in Table 4. These results are fairly tentative and should be regarded with some caution. For one, the column named *txt* reports performance of <desc> and <narr> fields which have been processed with our suffix-trimmer. This means some NLP has been done already (tagging + lexicon), and therefore what we see there is not the performance of 'pure' statistical system. The same applies to *con* column. (For

[14] For CACM-3204 collection the filter was most effective at $\sigma = 0.57$. For TREC-1 we changed the similarity formula slightly in order to obtain correct normalizations in all cases. This however lowered similarity coefficients in general and a new threshold had to be selected. We used $\sigma = 0.1$ in TREC-1 runs, although it turned out to be a poor choice. In all cases δ varied between 10 and 100.

[15] Bruce Croft (personal communication, 1992) has suggested that excluding all expert-made fields (i.e., <con> and <fac>) would make the queries quite ineffective. Broglio (personal communication, 1993) confirms this showing that text-only retrieval (i.e., with <desc> and <narr>) shows an average precision at more than 30% below that of <con>-based retrieval.

word1	word2	SIMnorm
abm	**anti+ballistic*	0.534894
absence	**maternity*	0.233082
accept	*acquire*	0.179078
accord	*pact*	0.492332
acquire	*purchase*	0.449362
speech	*address*	0.263789
adjustable	*one+year*	0.824053
maxsaver	**advance+purchase*	0.734008
affair	*scandal*	0.684877
affordable	*low+income*	0.181795
disease	**ailment*	0.247382
medium+range	**air+to+air*	0.874508
aircraft	**jetliner*	0.166777
aircraft	*plane*	0.423831
airline	*carrier*	0.345490
alien	*immigrate*	0.270412
anniversary	**bicentennial*	0.588210
anti+age	*anti+wrinkle*	0.153918
anti+clot	*cholesterol+lower*	0.856712
contra	**anti+sandinista*	0.294677
candidate	**aspirant*	0.116025
contend	**aspirant*	0.143459
property	*asset*	0.285299
attempt	*bid*	0.641592
await	*pend*	0.572960
stealth	**b+1*	0.877582
child	**baby*	0.183064
baggage	*luggage*	0.607333
ban	*restrict*	0.321943
bearish	*bullish*	0.847103
bee	**honeybee*	0.461023
roller+coast	**bumpy*	0.898278
two+income	*two+earner*	0.293104
television	*tv*	0.806018
soldier	*troop*	0.374410
treasury	**short+term*	0.661133
research	*study*	0.209257
withdrawal	**pullout*	0.622558

Table 1. Selecte filtered word similarities (* indicates the more specific term).

word	cluster
takeover	*merge, buy-out acquisition, bid*
stock	*share, issue, bond, price*
staff	*personnel, employee, force*
share	*stock, issue, fund*
sensitive	*crucial, difficult, critical*
rumor	*speculate*
president	*director, executive chairman, manage*
outlook	*forecast, prospect trend, picture*
law	*rule, legislate bill, regulate*
earnings	*revenue, income*
portfolio	*asset, invest, loan property, hold*
inflate	*growth, earnings, rise*
industry	*business, company, market*
help	*additional, support, involve*
growth	*increase, rise, gain decline, earnings, profit*
firm	*bank, concern, group, unit*
environ	*climate, condition situation, trend*
debt	*loan, secure, bond*
custom(er)	*client, investor buyer, consume(r)*
counsel	*attorney*
compute	*machine, software*
competitor	*rival, partner, buyer*
company	*business, firm, bank market, industry, concern*
big	*large, major, huge*
base	*facile, source reserve, support*
asset	*property, loan, fund, invest share, stock, money*

Table 2. Selected clusters obtained from approx. 10^7 words of text with weighted Tanimoto formula.

comparison, see Table 3 where runs with CACM-3204 collection included 'pure' statistics run (base), and note the impact our suffix trimmer is having.) Nonetheless, one may notice that automated NLP can be very effective at discovering the right query from an imprecise narrative specification: as much as 82% of the effectiveness of the expert-generated query can be attained.

CONCLUSIONS

We presented in some detail a natural language information retrieval system consisting of an advanced NLP module and a 'pure' statistical core engine. While many problems remain to be resolved, including the question of adequacy of term-based representation of document contents, we attempted to demonstrate that the architecture described here is nonetheless viable. In particular, we demonstrated that natural language processing can now be done on a fairly large scale and that its speed and robustness can match those of traditional statistical programs such as key-word indexing or statistical phrase extraction. We suggest, with some caution until more experiments are run, that natural language processing can be very effective in creating appropriate search queries out of user's initial specifications which can be frequently imprecise or vague.

On the other hand, we must be aware of the limits of NLP technologies at our disposal. While part-of-speech tagging, lexicon-based stemming, and parsing can be done on large amounts of text (hundreds of millions of words and more), other, more advanced processing involving conceptual structuring, logical forms, etc., is still beyond reach, computationally. It may be assumed that these super-advanced techniques will prove even more effective, since they address the problem of representation-level limits, however the experimental evidence is sparse and necessarily limited to rather small scale tests (e.g., Mauldin, 1991).

ACKNOWLEDGEMENTS

We would like to thank Donna Harman of NIST for making her PRISE system available to us. We would also like to thank Ralph Weischedel and Heidi Fox of BBN for providing and assisting in the use of the part of speech tagger. Jose Perez Carballo has contributed a number of valuable observations during the course of this work, and his assistance in processing the TREC data was critical. This paper is based upon work supported by the Defense Advanced Research Project Agency under Contract N00014-90-J-1851 from the Office of Naval Research, under Contract N00600-88-D-3717 from PRC Inc., and the National Science Foundation under Grant IRI-89-02304. We also acknowledge support from Canadian Institute for Robotics and Intelligent Systems (IRIS).

REFERENCES

Broglio, John and W. Bruce Croft. 1993. "Query Processing for Retrieval from Large Text Bases." Proceedings of ARPA HLT Workshop, March 21-24, Plainsboro, NJ.

Church, Kenneth Ward and Hanks, Patrick. 1990. "Word association norms, mutual information, and lexicography." *Computational Linguistics*, 16(1), MIT Press, pp. 22-29.

Crouch, Carolyn J. 1988. "A cluster-based approach to thesaurus construction." Proceedings of ACM SIGIR-88, pp. 309-320.

Grefenstette, Gregory. 1992. "Use of Syntactic Context To Produce Term Association Lists for Text Retrieval." Proceedings of SIGIR-92, Copenhagen, Denmark. pp. 89-97.

Grishman, Ralph, Lynette Hirschman, and Ngo T. Nhan. 1986. "Discovery procedures for sublanguage selectional patterns: initial experiments". *Computational Linguistics*, 12(3), pp. 205-215.

Grishman, Ralph and Tomek Strzalkowski. 1991. "Information Retrieval and Natural Language Processing." Position paper at the workshop on Future Directions in Natural Language Processing in Information Retrieval, Chicago.

Harman, Donna. 1988. "Towards interactive query expansion." Proceedings of ACM SIGIR-88, pp. 321-331.

Harman, Donna and Gerald Candela. 1989. "Retrieving Records from a Gigabyte of text on a Minicomputer Using Statistical Ranking." *Journal of the American Society for Information Science*, 41(8), pp. 581-589.

Hindle, Donald. 1990. "Noun classification from predicate-argument structures." Proc. 28 Meeting of the ACL, Pittsburgh, PA, pp. 268-275.

Lewis, David D. and W. Bruce Croft. 1990. "Term Clustering of Syntactic Phrases". Proceedings of ACM SIGIR-90, pp. 385-405.

Mauldin, Michael. 1991. "Retrieval Performance in Ferret: A Conceptual Information Retrieval System." Proceedings of ACM SIGIR-91, pp. 347-355.

Meteer, Marie, Richard Schwartz, and Ralph Weischedel. 1991. "Studies in Part of Speech Labeling." Proceedings of the 4th DARPA Speech and Natural Language Workshop,

Morgan-Kaufman, San Mateo, CA. pp. 331-336.
Sager, Naomi. 1981. *Natural Language Information Processing*. Addison-Wesley.
Sparck Jones, Karen. 1972. "Statistical interpretation of term specificity and its application in retrieval." *Journal of Documentation*, 28(1), pp. 11-20.
Sparck Jones, K. and E. O. Barber. 1971. "What makes automatic keyword classification effective?" *Journal of the American Society for Information Science*, May-June, pp. 166-175.
Sparck Jones, K. and J. I. Tait. 1984. "Automatic search term variant generation." *Journal of Documentation*, 40(1), pp. 50-66.
Strzalkowski, Tomek and Barbara Vauthey. 1991. "Fast Text Processing for Information Retrieval." Proceedings of the 4th DARPA Speech and Natural Language Workshop, Morgan-Kaufman, pp. 346-351.
Strzalkowski, Tomek and Barbara Vauthey. 1992. "Information Retrieval Using Robust Natural Language Processing." Proc. of the 30th ACL Meeting, Newark, DE, June-July. pp. 104-111.
Strzalkowski, Tomek. 1992. "TTP: A Fast and Robust Parser for Natural Language." Proceedings of the 14th International Conference on Computational Linguistics (COLING), Nantes, France, July 1992. pp. 198-204.

Runs	*base*	*suff.trim*	*query exp.*
Recall	*Precision Averages*		
0.00	0.764	0.775	0.793
0.10	0.674	0.688	0.700
0.20	0.547	0.547	0.573
0.30	0.449	0.479	0.486
0.40	0.387	0.421	0.421
0.50	0.329	0.356	0.372
0.60	0.273	0.280	0.304
0.70	0.198	0.222	0.226
0.80	0.146	0.170	0.174
0.90	0.093	0.112	0.114
1.00	0.079	0.087	0.090
3-pt Avg.	0.328	0.356	0.371
%chg		+8.3	+13.1

Table 3. Run statistics for CACM-3204 database: with no NLP; with suffix trimmer; and with both phrases and similarities.

Run	txt	txt+nlp	con	con+nlp
Queries	50	50	50	50
Tot. number of docs over all queries				
Ret	9980	9980	9788	9975
Rel	6228	6228	6228	6228
RelRet	1598	1835	1927	2062
%chg		+14.8	+20.6	+29.0
Recall	Precision Averages			
0.00	0.6420	0.6917	0.7021	0.7539
0.10	0.3727	0.4194	0.4476	0.4848
0.20	0.2476	0.2959	0.3353	0.3641
0.30	0.1543	0.2150	0.2202	0.2674
0.40	0.1093	0.1513	0.1443	0.1735
0.50	0.0611	0.0959	0.0851	0.1001
0.60	0.0298	0.0396	0.0403	0.0665
0.70	0.0160	0.0175	0.0187	0.0103
0.80	0.0046	0.0047	0.0048	0.0024
0.90	0.0000	0.0027	0.0000	0.0010
1.00	0.0000	0.0000	0.0000	0.0010
Average Precisions				
11-pt	0.1489	0.1758	0.1817	0.2023
%chg		+18.0	+22.0	+35.8
3-pt	0.1044	0.1322	0.1417	0.1555
%chg		+26.6	+35.7	+48.9
at 5	0.4360	0.5000	0.4680	0.4800
%chg		+14.6	+7.3	+10.0
at 15	0.3453	0.3827	0.3880	0.4107
%chg		+10.8	+12.3	+18.9
at 100	0.2108	0.2384	0.2498	0.2712
%chg		+13.0	+18.5	+28.6

Table 4. Run statistics with TIPSTER WSJ database with top 200 documents considered per each query: (1) *txt* - with <narr> and <desc> fields only; (2) *txt+nlp* - with <narr> and <desc> only including syntactic phrase terms and similarities; (3) *con* - with <desc> and <con> fields only; and (4) *con+nlp* - with <desc> and <con> fields including phrases and similarities. In all cases documents preprocessed with lexicon-based suffix-trimmer.